

LA-UR-13-23154

Approved for public release; distribution is unlimited.

Title:	CBTF Tools Demo
Author(s):	Mason, Michael A. Montoya, David R.
Intended for:	Released to future users of CBTF. Web
Issued:	2013-05-01



Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

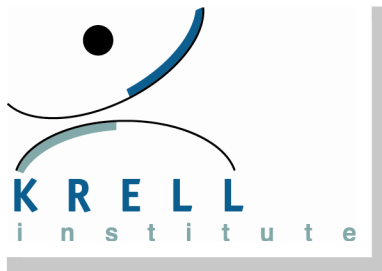
CBTF Tools Demo

Component Based Tool Framework Tools Demo

Mike Mason

Dave Montoya

LANL, HPC-3



❖ What is CBTF?

- A Framework for writing Tools that are Based on Components.

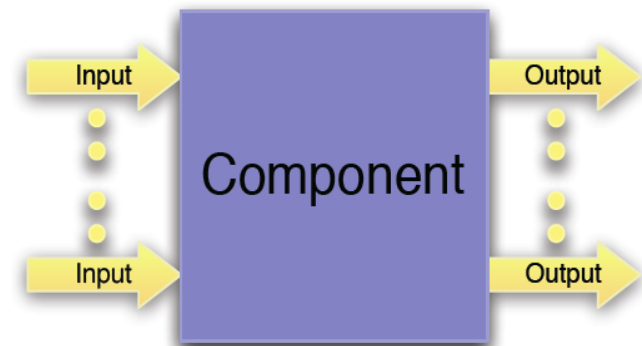
❖ Benefits of CBTF

- Components are reusable and easily added to new tools.
- With a large component repository new tools can be written quickly with little code.
- Tools automatically adjust to new topologies.
- Tools are inherently scalable.

❖ Originally made for Performance Analysis with Open | SpeedShop

- We look at alternative uses like sysadmin or monitoring tools.

- ❖ Reusable objects with 0-N inputs and 0-M outputs.
- ❖ Uses the dataflow programming model where each component is a black box that takes some input, performs a task then produces some output.
- ❖ Designed to be connected together
 - Connections defined in C++ or XML file.
- ❖ Components are written in C++
 - Components can do anything your C++ code can do.
 - Run system commands, open files, do calculations.



- ❖ **CBTF uses another mechanism to transport all of its communications.**
- ❖ **Right now that mechanism is MRNet**
 - Multicast/Reduction Network
 - Tree structure lends itself to improved scalability
- ❖ **CBTF view MRNet as just another component.**
 - In the future it can be easily swapped with some other transport mechanism.
 - It can be used within a component network already running on MRNet allowing for some control over the tree structure at runtime.

❖ Topology

- You must define a tree based topology with one root (Frontend) node, zero or more levels of communication process (Filter) nodes and some number of leaf (Backend) nodes.

❖ Normal mode (Backend Create)

- Both MRNet and CBTF components are started on the Backend automatically.

❖ Lightweight mode (Backend Attach)

- The MRNet tree is created but not on the Backend. An application is run separately on the Backend that attaches itself to the MRNet tree.

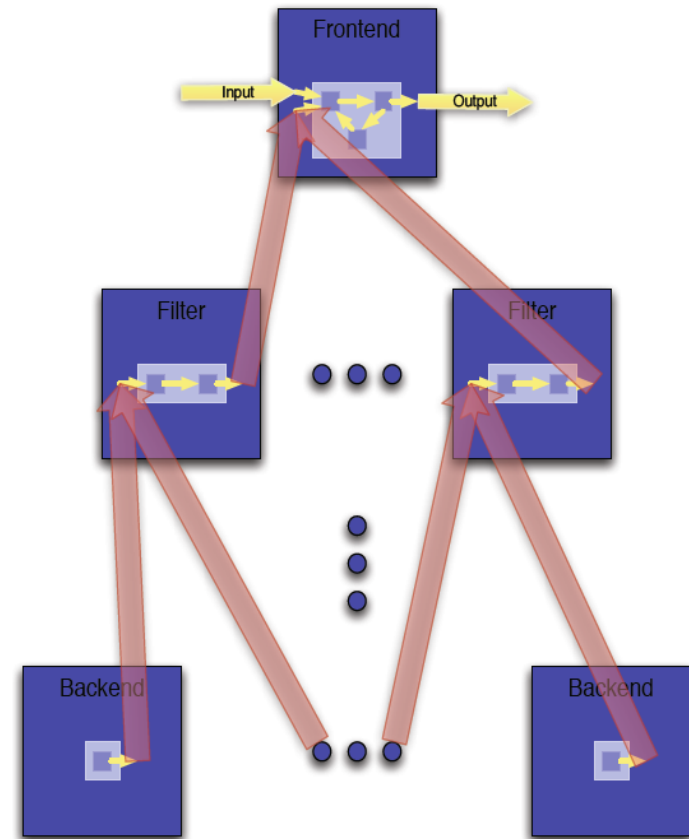
❖ Three Networks where components can be connected

- Frontend, Backend, multiple Filter levels
- Every level is homogeneous

❖ Each Network is also a component and therefore has some number of inputs and outputs.

❖ Any component can be run on any level, but logically

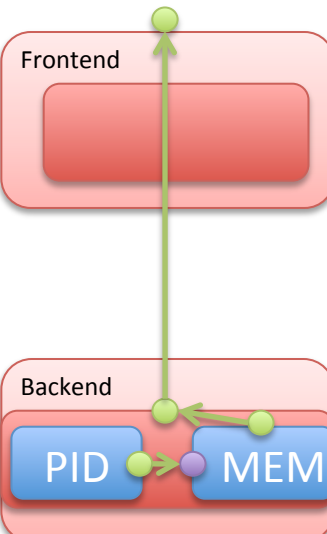
- Frontend
 - Interact with or Display info to the user
- Filter
 - Filter or Aggregate info from below
 - Make decisions about what is sent up or down the tree
- Backend
 - Real work of the tool



- ❖ **What can this framework be used for?**
- ❖ **CBTF is flexible and general enough to be used for any tool that needs to “do something” on a large number of nodes and filter or collect the results.**
- ❖ **Sysadmin Tools**
 - Poll information on a large number of nodes.
 - Run commands or manipulate files on the Backends.
 - Make decisions at the Filter level to reduce output or interaction
- ❖ **Performance Analysis Tools**
 - Massively parallel applications need scalable tools
 - Have components running along side the application
- ❖ **Debugging Tools**
 - Use cluster analysis to reduce thousands (or more) processes into a small number of groups

❖ memTool (tools/contrib/memTool)

- Displays memory info for each node and total memory used for each MPI process.



Frontend

Backend

PID

MEM

```
mmason@rra012a:memtest — ssh —
[mmason@rra012a memtest]$ mpirun -np 4 ./mpi_malloc
Reported: 1(1) daemons 4(4) procs
rank 0 started my_rank = 0
^Z
Suspended
[mmason@rra012a memtest]$ ps -u mmason
  PID TTY          TIME CMD
  8349 pts/0    00:00:00 tcsh
  8357 pts/0    00:00:00 pbs_mom
  8662 pts/0    00:00:00 mpirun
  8664 pts/0    00:00:09 mpi_malloc
  8665 pts/0    00:00:06 mpi_malloc
  8666 pts/0    00:00:09 mpi_malloc
  8667 pts/0    00:00:06 mpi_malloc
  8676 pts/0    00:00:00 ps_
```

BE

```
mmason@rr-dev-fe:bin — ssh — 8
[mmason@rr-dev-fe bin]$ ./memtool -be 1 mpi_malloc
tput: No value for $TERM and no -T specified
14325102.183208: FROMCHILD(rra012a:0)(0x439d9950): From
romChildren() - stream 1073741825 lookup failed
14325102.183889: FROMCHILD(rra012a:0)(0x439d9950): Par
FromChildren() - proc_DataFromChildren() failed
14325102.184009: FROMCHILD(rra012a:0)(0x439d9950): Pee
ain() - proc_PacketsFromChildren() failed
rra012a.rr.lanl.gov
```

	total	used	free	shared	buffers	cached
Mem:	16086	14960	1125	0	0	68

FE

```
mmason@rr-dev-fe:bin — ssh — 79x34
[mmason@rr-dev-fe bin]$ ./memtool -be 2 mpi_malloc
tput: No value for $TERM and no -T specified
tput: No value for $TERM and no -T specified
14325400.022360: FROMCHILD(rra011a:1)(0x42b00950): FrontEndNode.C[52] proc_Data
FromChildren() - stream 1073741825 lookup failed
14325400.023040: FROMCHILD(rra011a:1)(0x42b00950): ParentNode.C[129] proc_Packe
tFromChildren() - stream 1073741825 lookup failed
14325400.023138: FROMCHILD(rra011a:1)(0x42b00950): PeerNode.C[264] recv_thread_
main() - proc_PacketsFromChildren() failed
14325400.104697: FROMCHILD(rra010a:0)(0x43f02950): FrontEndNode.C[52] proc_Data
FromChildren() - stream 1073741825 lookup failed
14325400.104832: FROMCHILD(rra010a:0)(0x43f02950): ParentNode.C[129] proc_Packe
tFromChildren() - proc_DataFromChildren() failed
14325400.104955: FROMCHILD(rra010a:0)(0x43f02950): PeerNode.C[264] recv_thread_
main() - proc_PacketsFromChildren() failed
rra011a.rr.lanl.gov
```

	total	used	free	shared	buffers	cached
Mem:	16086	15001	1084	0	0	99

FE

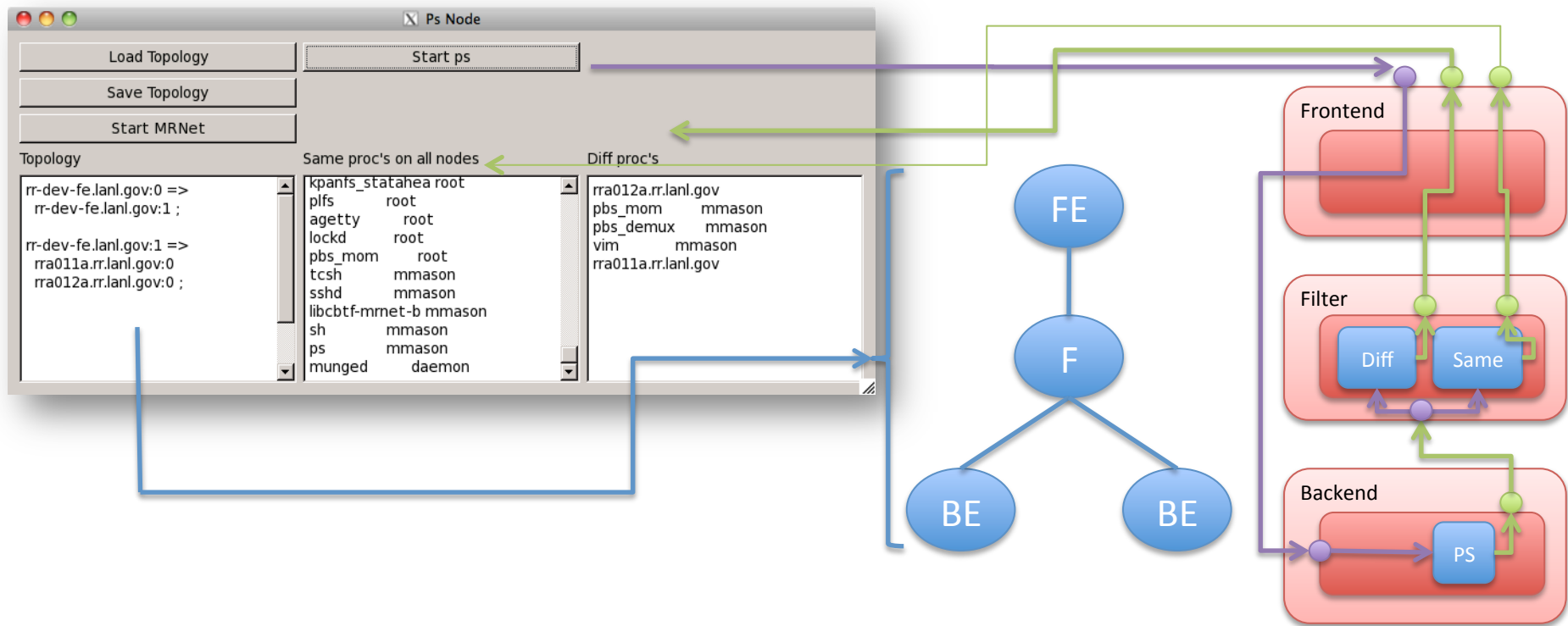
```
rra010a.rr.lanl.gov
total      used      free      shared      buffers      cached
Mem:       16086      15247      838        0          0          50
pid        %mem     mem(MB)
5061       23.7    3812.3820(MB)
5062       23.7    3812.3820(MB)
5063       21.3    3426.3178(MB)
5064       23.7    3812.3820(MB)
```

FE

```
[mmason@rr-dev-fe bin]$
```

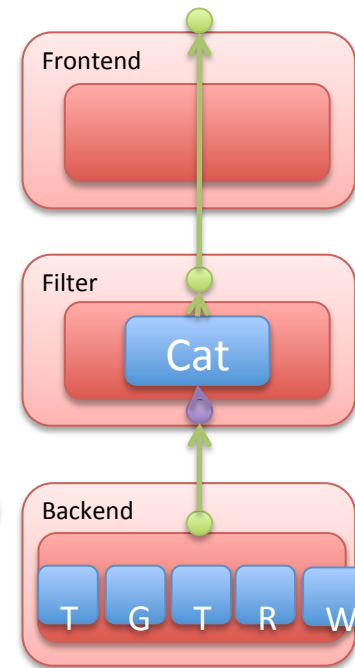
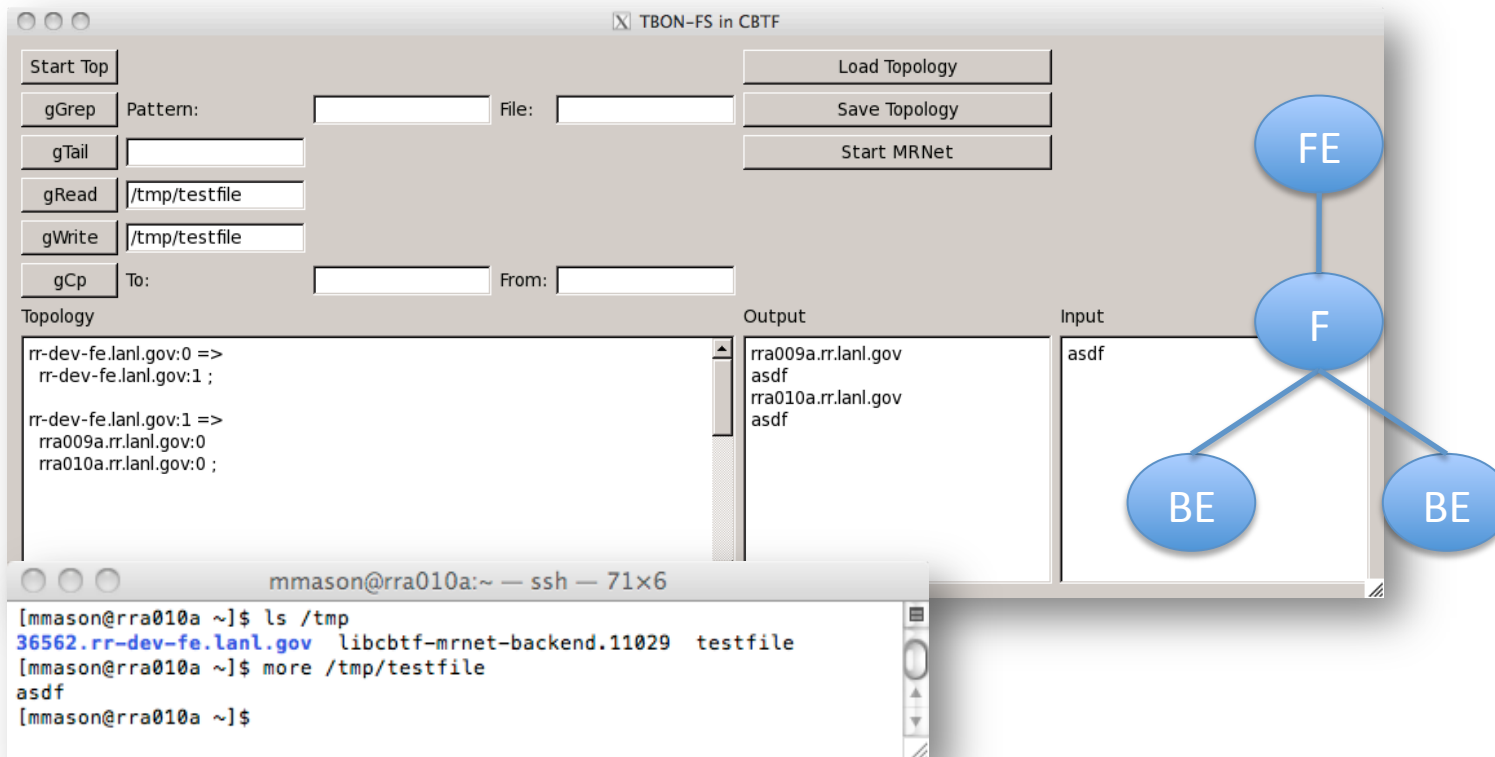
❖ PS Tool (tools/contrib/psTool)

- Run ps on all Backends, use filters to identify the procs running on all nodes and what procs are unique to each node.



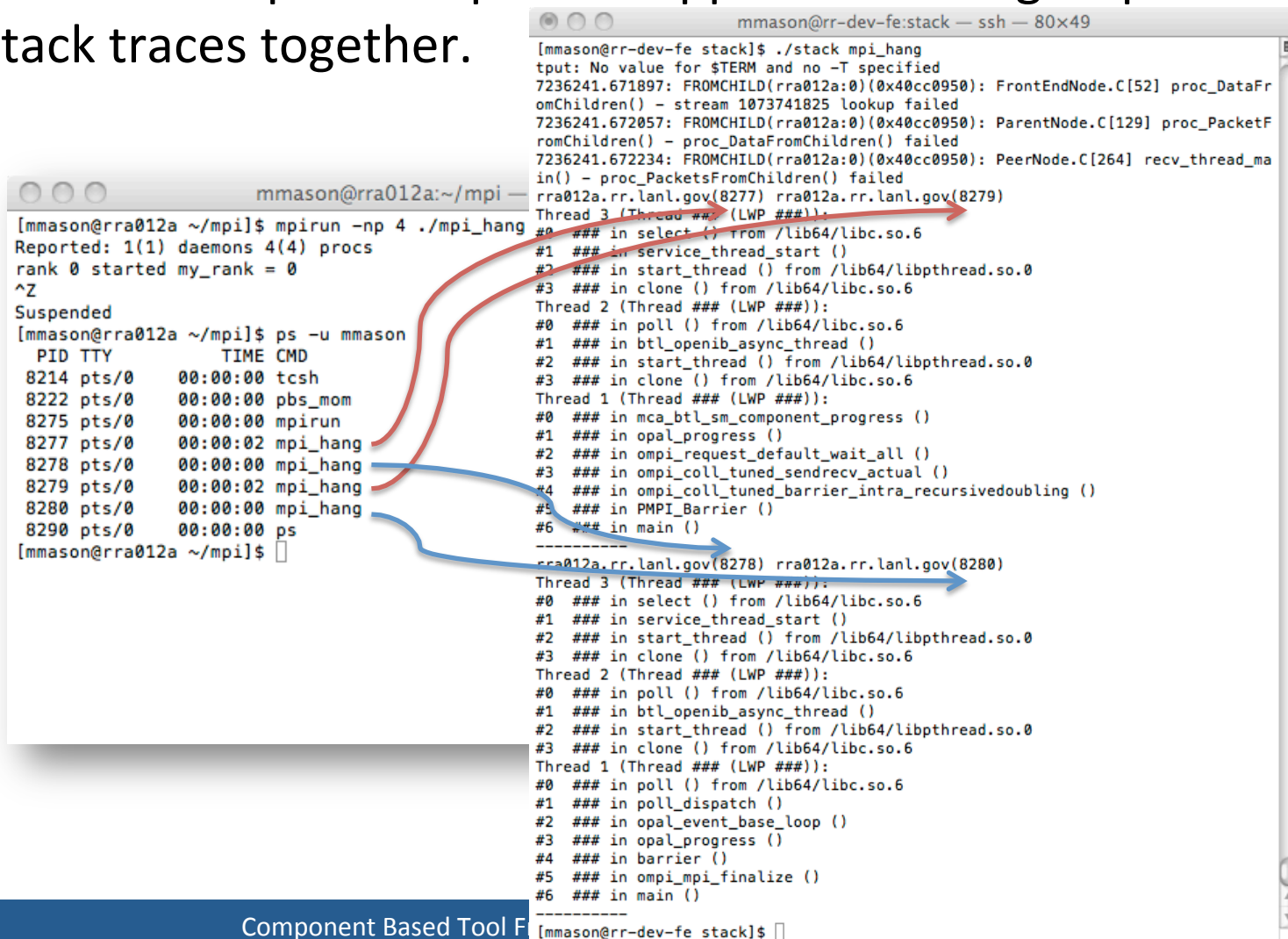
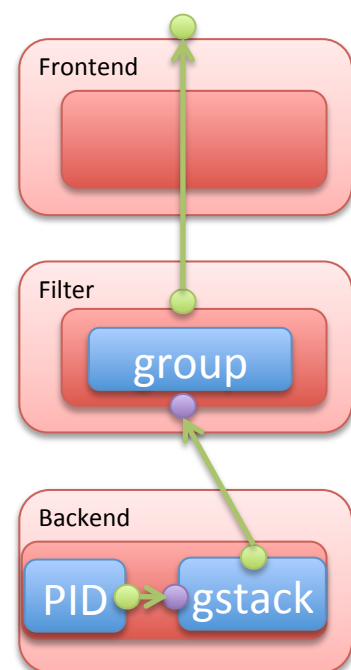
❖ TBON-FS implementation (tools/contrib/tbonFS)

- Perform group file operations on the Backend local file system.
- Run top on all Backends.
- Run grep, tail, read or write files on all Backends.
- Copy a file from NFS to the local /tmp on all Backends.



❖ Stack trace grouping (tools/contrib/stack)

- Run gstack on each pid for a parallel application and group similar stack traces together.



The screenshot shows two terminal windows. The left window, titled "mmason@rra012a:~/mpi", shows the execution of `mpirun -np 4 ./mpi_hang`, which reports a hang and lists the PIDs of the processes. The right window, titled "mmason@rr-dev-fe:stack", shows the output of `./stack mpi_hang`, which displays the stack traces for the hanging processes. Red and blue arrows connect the PIDs in the left window to their corresponding stack traces in the right window.

```
[mmason@rra012a ~/mpi]$ mpirun -np 4 ./mpi_hang
Reported: 1(1) daemons 4(4) procs
rank 0 started my_rank = 0
^Z
Suspended
[mmason@rra012a ~/mpi]$ ps -u mmason
  PID TTY          TIME CMD
  8214 pts/0    00:00:00 tcsh
  8222 pts/0    00:00:00 pbs_mom
  8275 pts/0    00:00:00 mpirun
  8277 pts/0    00:00:02 mpi_hang
  8278 pts/0    00:00:00 mpi_hang
  8279 pts/0    00:00:02 mpi_hang
  8280 pts/0    00:00:00 mpi_hang
  8290 pts/0    00:00:00 ps
[mmason@rra012a ~/mpi]$
```

```
[mmason@rr-dev-fe stack]$ ./stack mpi_hang
tput: No value for $TERM and no -T specified
7236241.671897: FROMCHILD(rra012a:0)(0x40cc0950): FrontEndNode.C[52] proc_DataFromChildren() - stream 1073741825 lookup failed
7236241.672057: FROMCHILD(rra012a:0)(0x40cc0950): ParentNode.C[129] proc_PacketFromChildren() - proc_DataFromChildren() failed
7236241.672234: FROMCHILD(rra012a:0)(0x40cc0950): PeerNode.C[264] recv_thread_main() - proc_PacketsFromChildren() failed
rra012a.rr.lanl.gov(8277) rra012a.rr.lanl.gov(8279)
Thread 3 (Thread ### (LWP ###)):
#0  ### in select () from /lib64/libc.so.6
#1  ### in service_thread_start ()
#2  ### in start_thread () from /lib64/libpthread.so.0
#3  ### in clone () from /lib64/libc.so.6
Thread 2 (Thread ### (LWP ###)):
#0  ### in poll () from /lib64/libc.so.6
#1  ### in btl_openib_async_thread ()
#2  ### in start_thread () from /lib64/libpthread.so.0
#3  ### in clone () from /lib64/libc.so.6
Thread 1 (Thread ### (LWP ###)):
#0  ### in mca_btl_sm_component_progress ()
#1  ### in opal_progress ()
#2  ### in ompi_request_default_wait_all ()
#3  ### in ompi_coll_tuned_sendrecv_actual ()
#4  ### in ompi_coll_tuned_barrier_intra_recursivedoubling ()
#5  ### in PMPI_Barrier ()
#6  ### in main ()
-----
rra012a.rr.lanl.gov(8278) rra012a.rr.lanl.gov(8280)
Thread 3 (Thread ### (LWP ###)):
#0  ### in select () from /lib64/libc.so.6
#1  ### in service_thread_start ()
#2  ### in start_thread () from /lib64/libpthread.so.0
#3  ### in clone () from /lib64/libc.so.6
Thread 2 (Thread ### (LWP ###)):
#0  ### in poll () from /lib64/libc.so.6
#1  ### in btl_openib_async_thread ()
#2  ### in start_thread () from /lib64/libpthread.so.0
#3  ### in clone () from /lib64/libc.so.6
Thread 1 (Thread ### (LWP ###)):
#0  ### in poll () from /lib64/libc.so.6
#1  ### in poll_dispatch ()
#2  ### in opal_event_base_loop ()
#3  ### in opal_progress ()
#4  ### in barrier ()
#5  ### in ompi_mpi_finalize ()
#6  ### in main ()
-----
[mmason@rr-dev-fe stack]$
```

- ❖ **Collect data on a program running on multiple nodes and transmit that data using MRNet**
 - Statistics about the amount of memory being used can be gathered at both the individual node level and aggregated over every node (or a specific subset of nodes).
- ❖ **Uses the MRNet BackendAttach mode**
 - Backends trace memory functions such as malloc, calloc, realloc, and free via interposition
 - This allows the tool to keep track of possible memory leaks
- ❖ **Still a work in progress**

❖ Problems

- CBTF is still being written
- Very dependent on MRNet
 - Not easily changeable
 - Must know about MRNet and how to use it
- XML file gets big and complicated fast
 - At some point there will be a GUI viewer/builder

❖ Summary

- Provides a framework for writing scalable tools.
- Easily reuse and add components to tools.
- We can make simple example tools now
 - Not ready for the real world yet